Burp But Yours

Hands-on Extension and Bambda development



Hannah L

PortSwigger

Agenda

1	Welcome + setup	5 min
2	Introduction: Extensibility landscape, Montoya API	15 min
3	Bambdas: introduction and exercise	30 min
4	Extensions: introduction and exercise	30 min
5	Free build time	20 min
6	Sharing your work, wrap-up, Q&A	10 min

Aims and Objectives



Identify and explain extensibility options in Burp



Write a functional Bambda



Write a functional extension

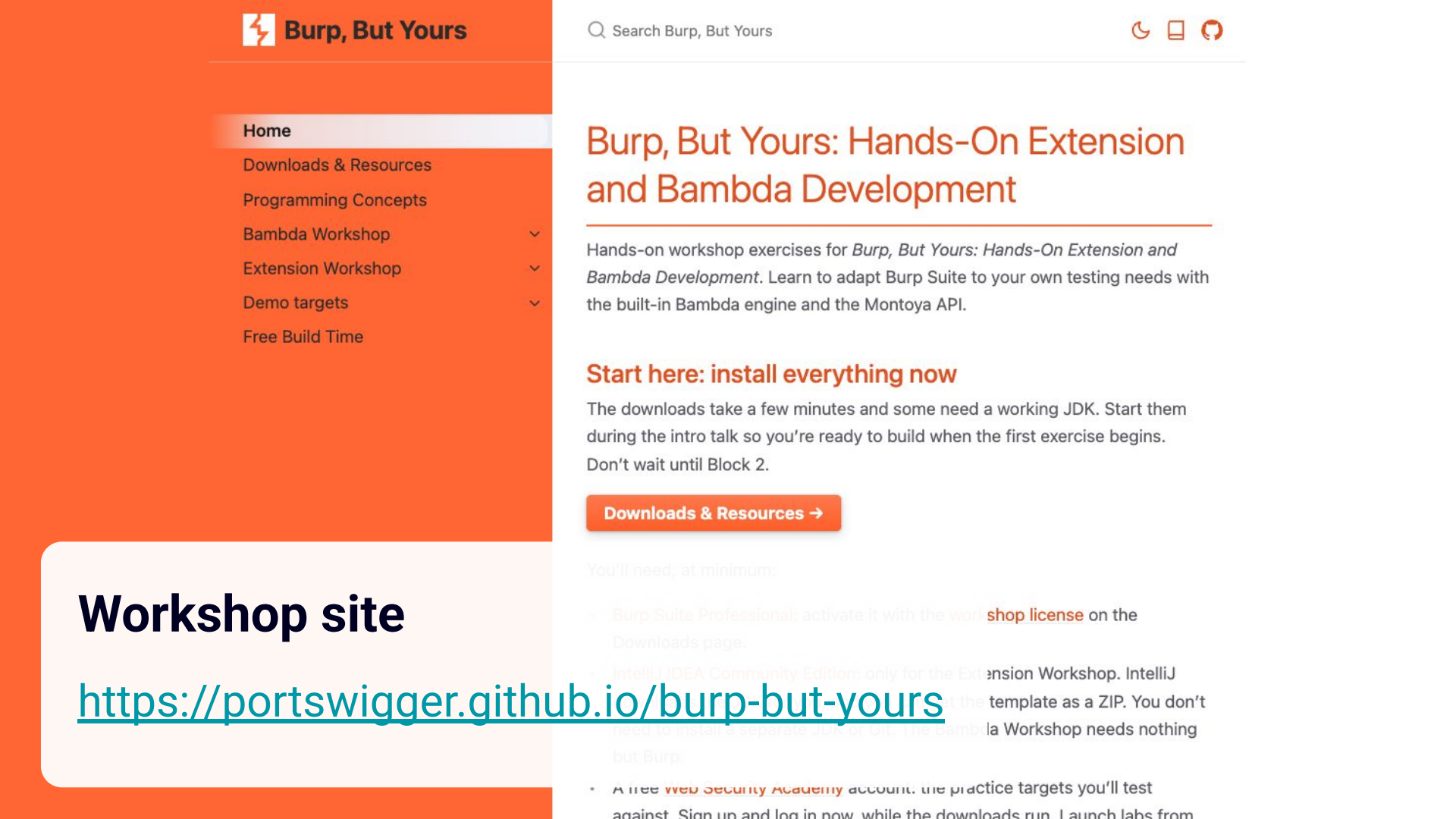


Know how to share your work more widely

You will need

Make sure you have these open, and have logged in where appropriate

- portswigger.net account for Web Security Academy labs
- IntelliJ IDEA, or other Java IDE, for development
- Latest version of Burp Suite downloaded
- Workshop site: https://portswigger.github.io/burp-but-yours/



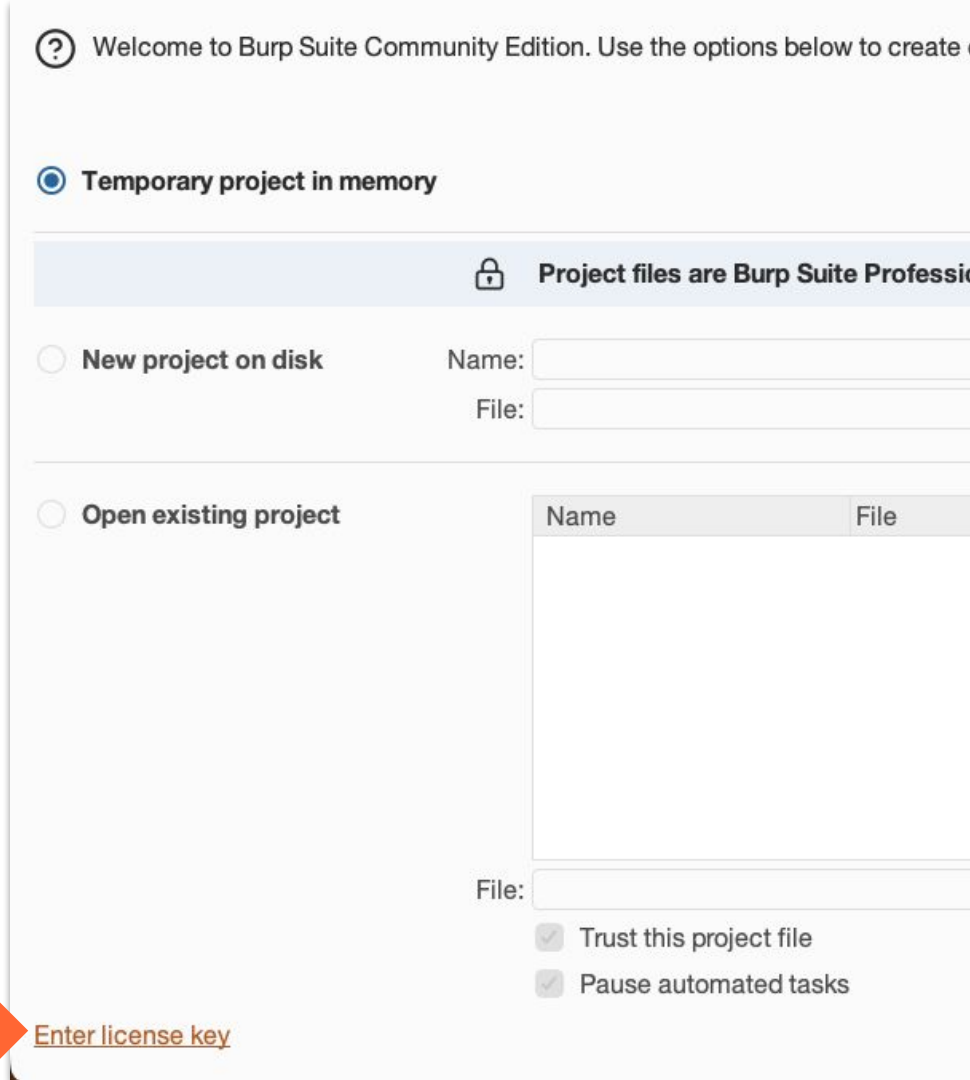
Workshop site

https://portswigger.github.io/burp-but-yours

Upgrade your Burp

Some of the features we'll use today are exclusive to Burp Suite Professional.

https://portswigger.github.io/burp-but-yours/ contains a short term training license. If you don't have Burp Suite Professional activated, apply this license.



Extensibility landscape

What is Burp Suite

Professional / Community

A collection of manual and semi-automated tools for web application penetration testing

Pro includes an automated Scanner and additional tooling

DAST

Enterprise-grade software for scanning web applications at scale

write down...

What is one repetitive manual testing task you'd automate if you could?

Why Extensibility?



The vulnerability landscape is evolving faster than ever



Every target has quirks that no off-the-shelf check will catch



Your testing workflow is yours - your tools should be too

What Extensibility options are there?



Custom scan checks



Bambda scripts



Extensions / BApps

Professional

DAST

Custom Scan Checks

BCheck scripting language

Custom simple scripting language

Built-in functions for handling common actions

Choose firing frequency - every host, path, request, or insertion point

Low complexity

Java

Write in Java

Access to the Montoya API and Java libraries

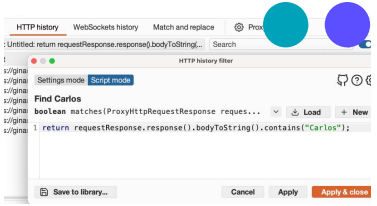
Choose firing frequency - every host, request, or insertion point

High complexity

Community

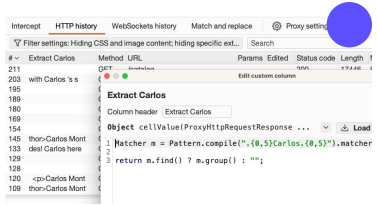
Professional

Bambda Scripts



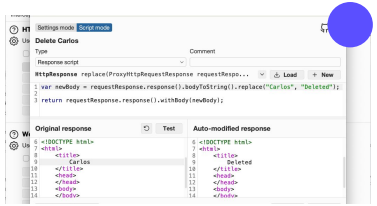
Filters

Pinpoint interesting requests/responses that meet your criteria



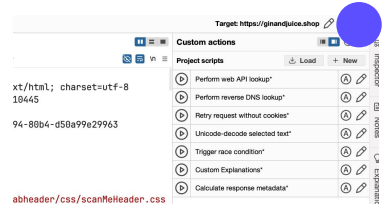
Custom columns

Extract and analyze data from requests and responses - surface relevant data quickly



Match & Replace

Target specific traffic in the Proxy for complex match and replace logic



Custom Actions

“Create your own Repeater feature” by extracting, analyzing, and manipulating requests

Extensions

Community
Professional
DAST

Extensions

Implement custom functionality that doesn't exist anywhere else

Hook into almost any part of Burp

Automate repetitive tasks

Extend Burp to cope with stuff no existing tool can handle

Build and load your extension into Burp immediately

BApps

An extension that has been through PortSwigger's review process, and published to the "BApp Store"

Available to all users of Burp Suite, depending on product compatibility

Easy, one-click install

Automatically stay up to date with the latest published BApp version

Before you create

Check community resources

BChecks PortSwigger/BChecks

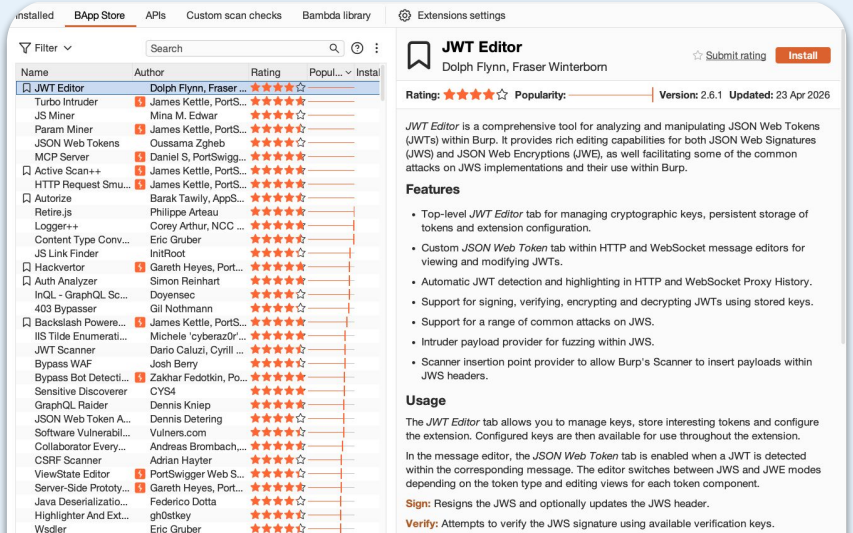
Bambdas PortSwigger/bambdas

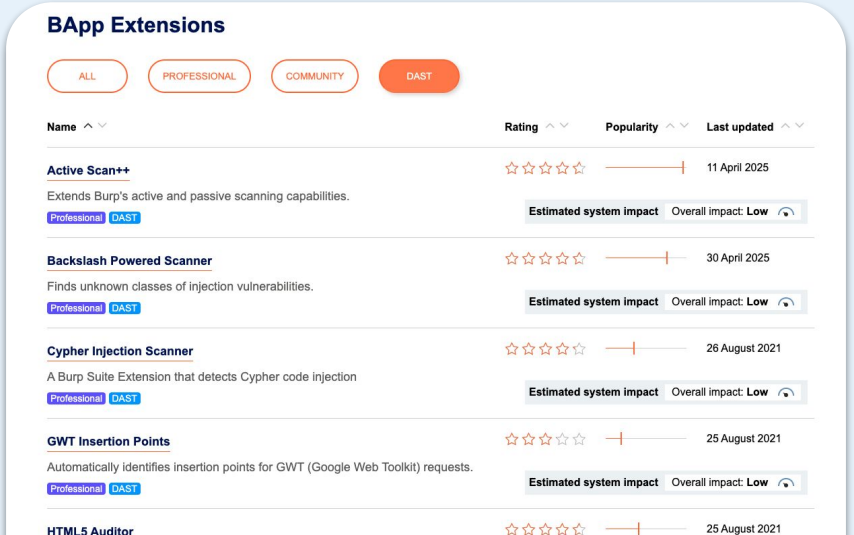
Extensions **BApp Store or GitHub**

Where is the BApp Store

Professional, Community: Extensions > BApp Store

https://portswigger.net/bappstore





Today’s focus

Java-based extensibility



Bambda Scripts



Extensions

Montoya API

Core concepts

Variable

Named value to store data

Class

A blueprint that defines fields and methods

Object

An instance of a class

Field

A variable that belongs to a class or object

Method

A function that belongs to a class or object

Interface

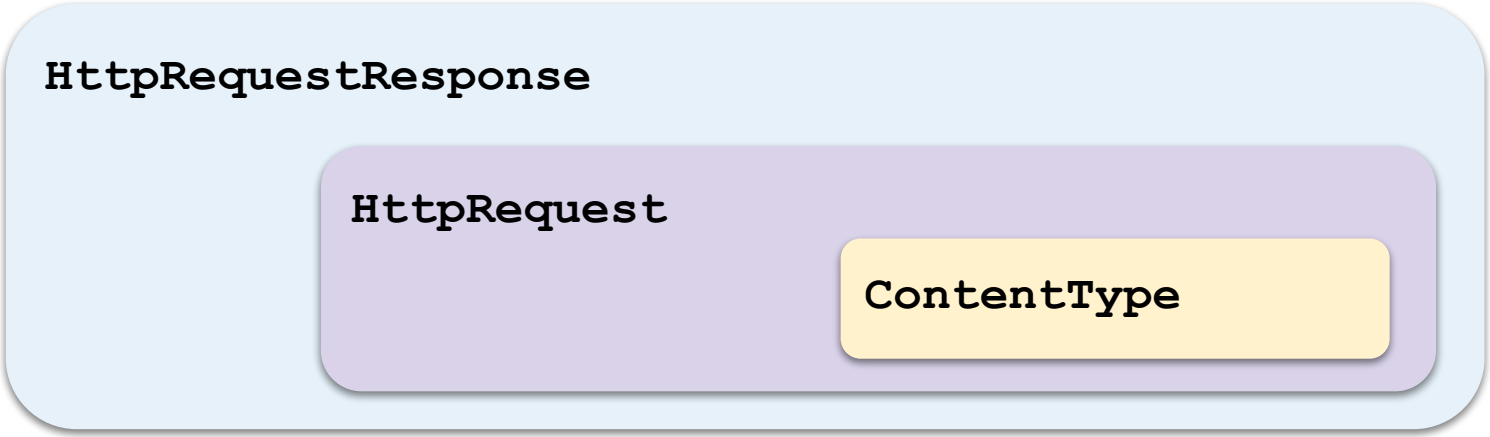
Defines what methods a class must provide

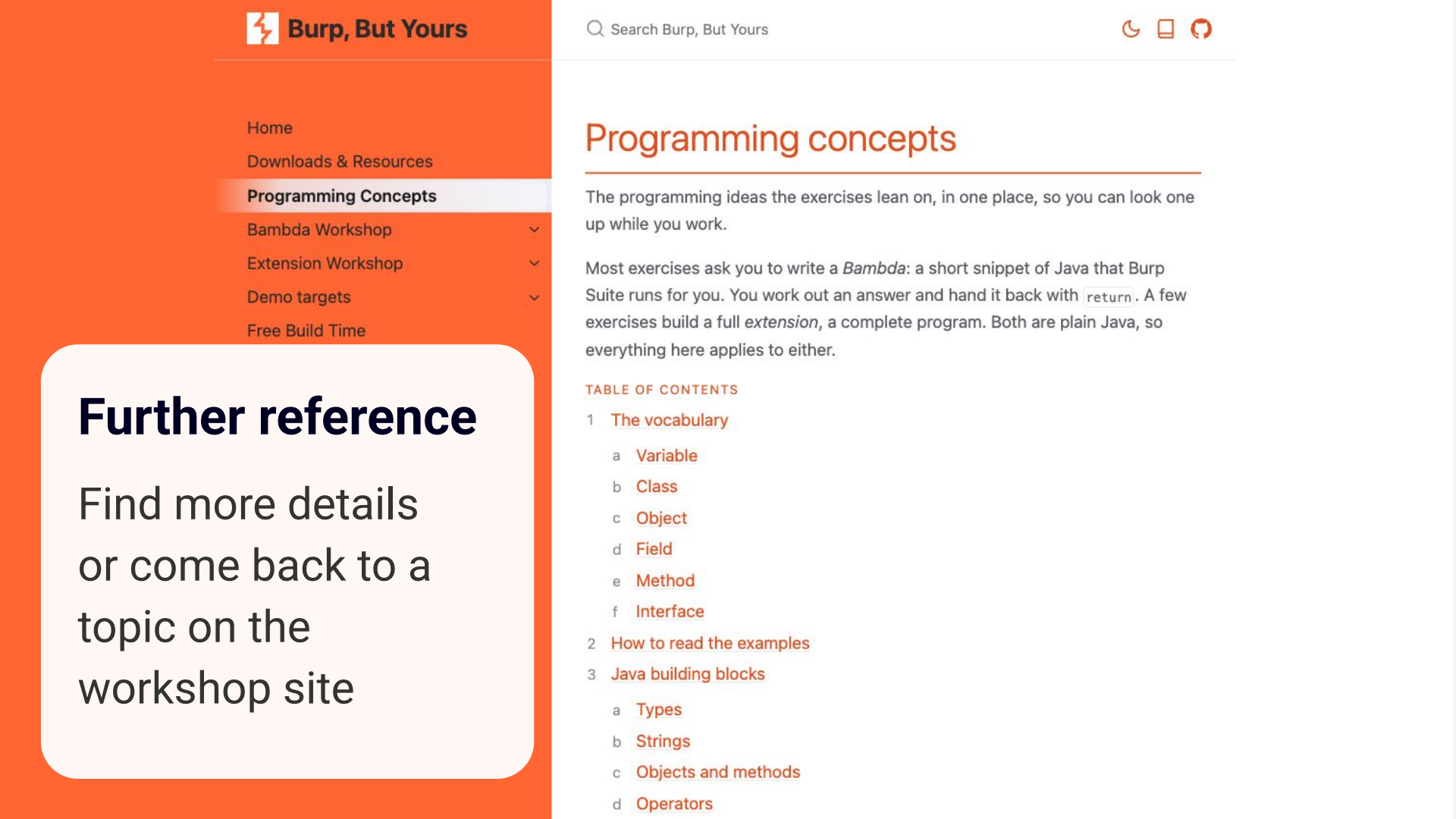
Method chaining

Calling one method after another in a single statement

Use this to transition from one object to another

e.g., `httpRequestResponse.request().contentType()`




Burp, But Yours
Search Burp, But Yours
Home
Downloads & Resources
Programming Concepts
Bambda Workshop
Extension Workshop
Demo targets
Free Build Time
Programming concepts
The programming ideas the exercises lean on, in one place, so you can look one up while you work.
Most exercises ask you to write a Bambda: a short snippet of Java that Burp Suite runs for you. You work out an answer and hand it back with return. A few exercises build a full extension, a complete program. Both are plain Java, so everything here applies to either.
TABLE OF CONTENTS
1 The vocabulary
a Variable
b Class
c Object
d Field
e Method
f Interface
2 How to read the examples
3 Java building blocks
a Types
b Strings
c Objects and methods
d Operators
Further reference
Find more details or come back to a topic on the workshop site

What is the Montoya API?

The entry point for your code to interact and hook into Burp

All Java code you use in Burp - whether writing Bambda scripts or extensions - will interact with this

Extensions have full access to the Montoya API

Bambda scripts have more limited access, depending on the type of Bambda you are writing

Interface MontoyaApi

`public interface MontoyaApi`

This interface is used by Burp Suite to pass a set of methods to extensions that can be used to perform various actions within Burp. When an extension is loaded, Burp invokes its `BurpExtension.initialize(MontoyaApi)` method and passes an instance of the `MontoyaApi` interface. The extension may then invoke the methods of this interface as required in order to extend Burp's functionality.

Method Summary

All Methods | Instance Methods | Abstract Methods

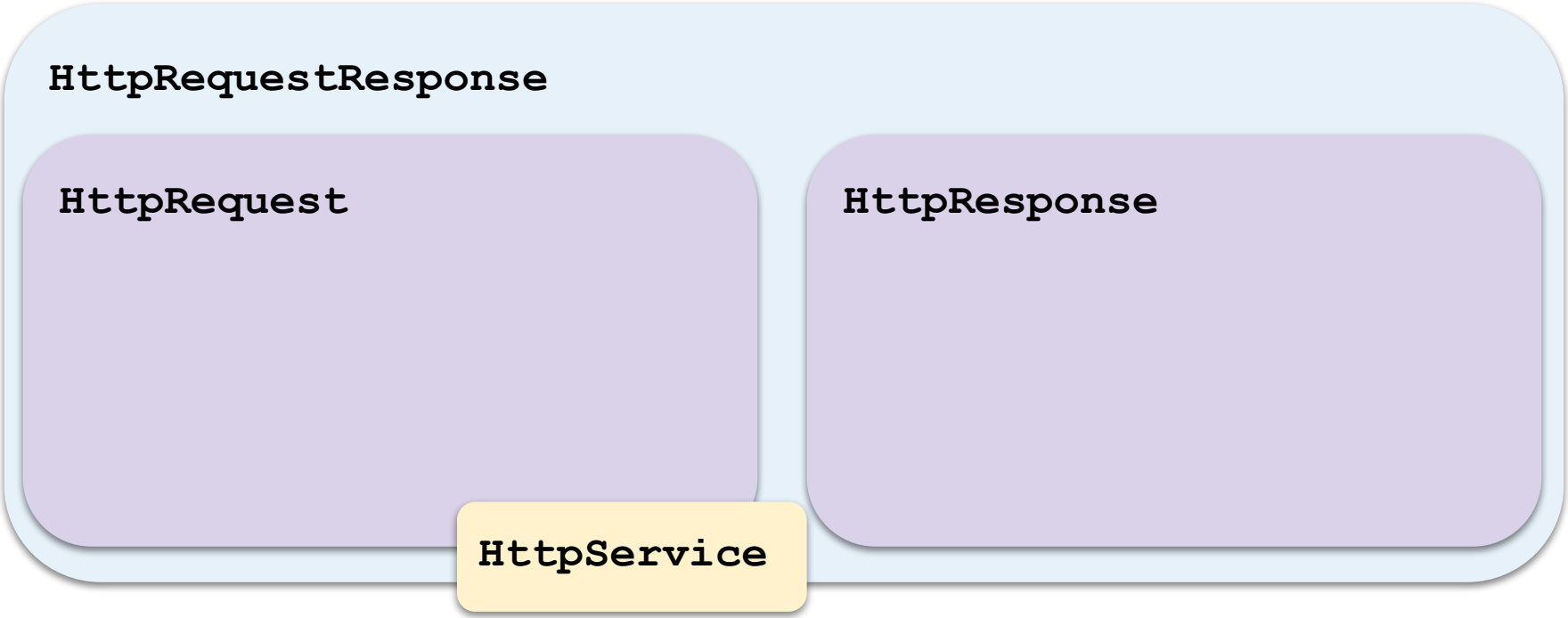
Modifier and Type	Method	Description
`Ai`	`ai()`	[Professional only] Access AI related functionality.
`Bambda`	`bambda()`	Access functionality related to Bambdas.
`BurpSuite`	`burpSuite()`	Access functionality related to the Burp Suite application.
`Collaborator`	`collaborator()`	[Professional only] Access the functionality of the Collaborator.
`Comparer`	`comparer()`	Access the functionality of the Comparer.
`Decoder`	`decoder()`	Access the functionality of the Decoder.
`Extension`	`extension()`	Access functionality related to your extension.
`Http`	`http()`	Access the functionality related to HTTP requests and responses.
`Intruder`	`intruder()`	Access the functionality of the Intruder.
`Logging`	`logging()`	Access the functionality related to logging and events.
`Organizer`	`organizer()`	Access the functionality of the Organizer.
`Persistence`	`persistence()`	Access the functionality related to persistence.
`Project`	`project()`	Access functionality related to the project.
`Proxy`	`proxy()`	Access the functionality of the Proxy.
`Repeater`	`repeater()`	Access the functionality of the Repeater.
`Scanner`	`scanner()`	[Professional only] Access the functionality of the Scanner.
`Scope`	`scope()`	Access the functionality related to Burp's suite-wide target scope.
`SiteMap`	`siteMap()`	Access the functionality of the Site Map.
`UserInterface`	`userInterface()`	Access the functionality related to the user interface.
`WebSockets`	`websockets()`	Access the functionality related to WebSockets and messages.

https://portswigger.github.io/burp-extensions-montoya-api/javadoc/burp/api/montoya/MontoyaApi.html

Key areas

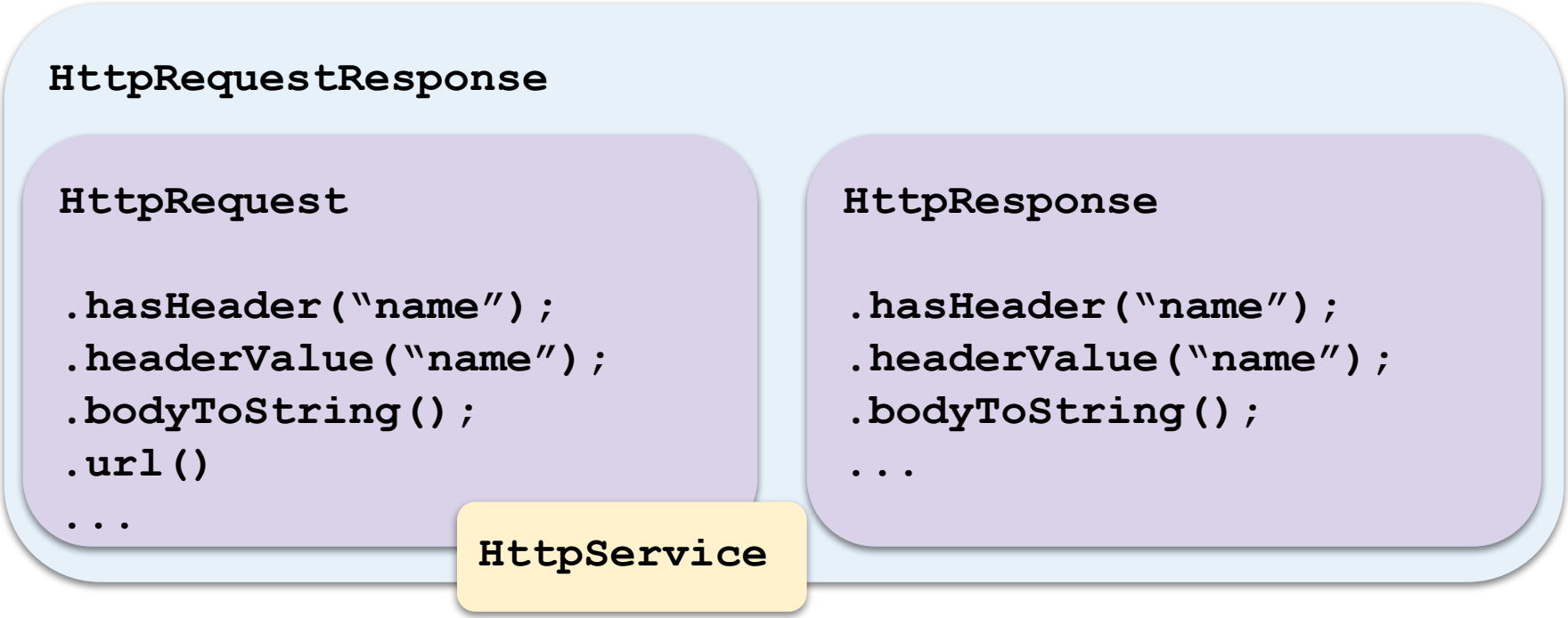
1. Working with requests and responses
2. Logging
3. Utility functions

HttpRequestResponse



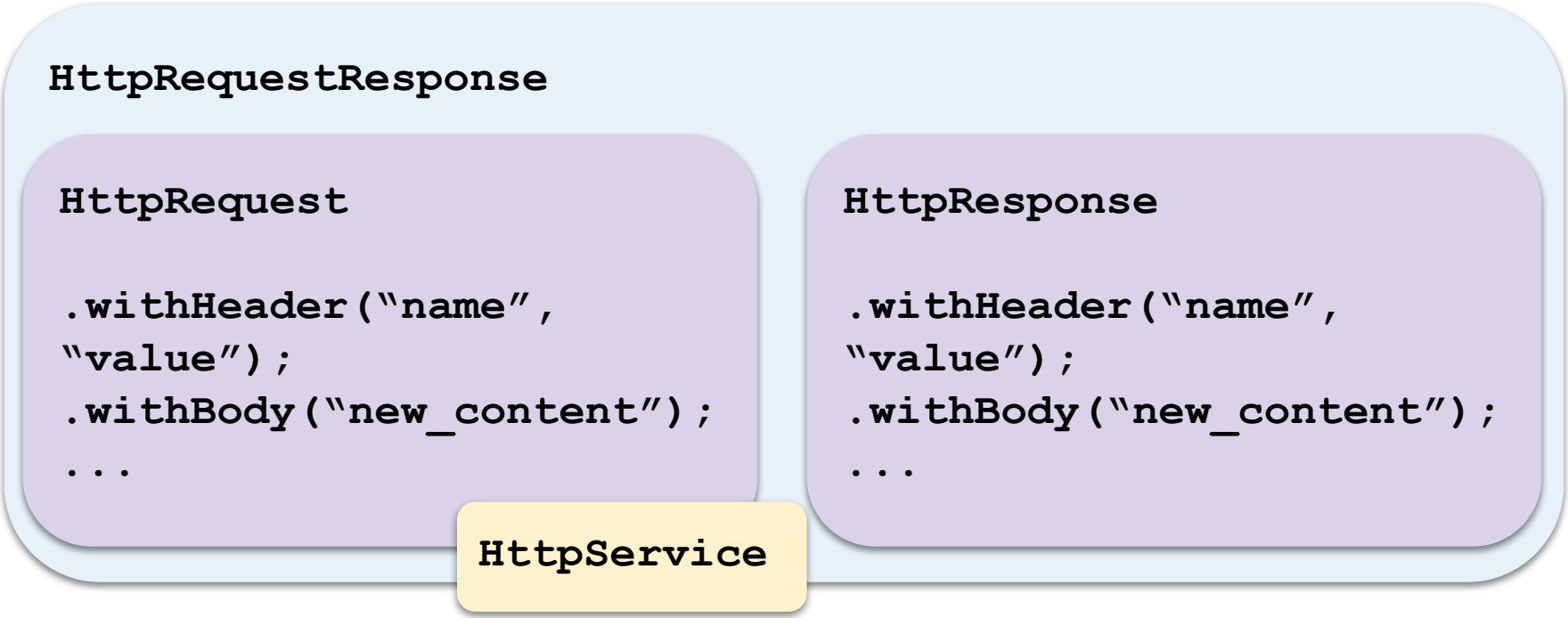

```
HttpRequest request = requestResponse.request();
```

Reading data from a request or response



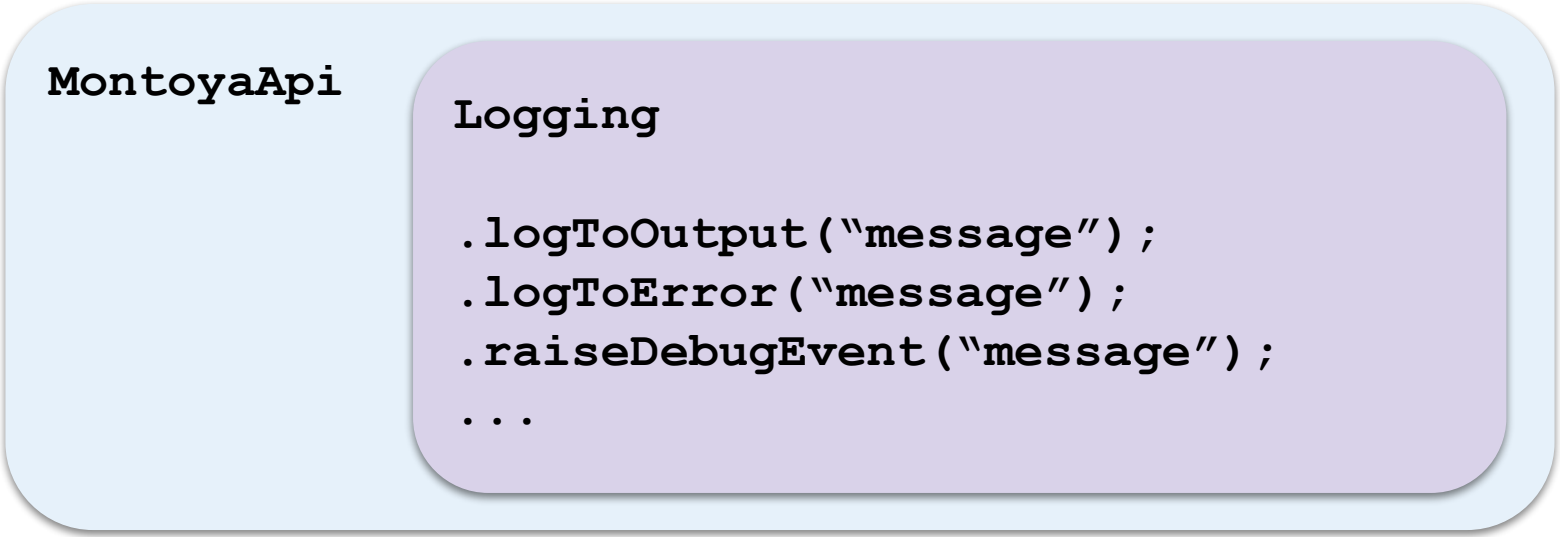

```
String userAgent = request.headerValue("User-Agent");
```

Manipulating requests and responses




```
HttpRequest modified = ogRequest.withHeader(...);
```

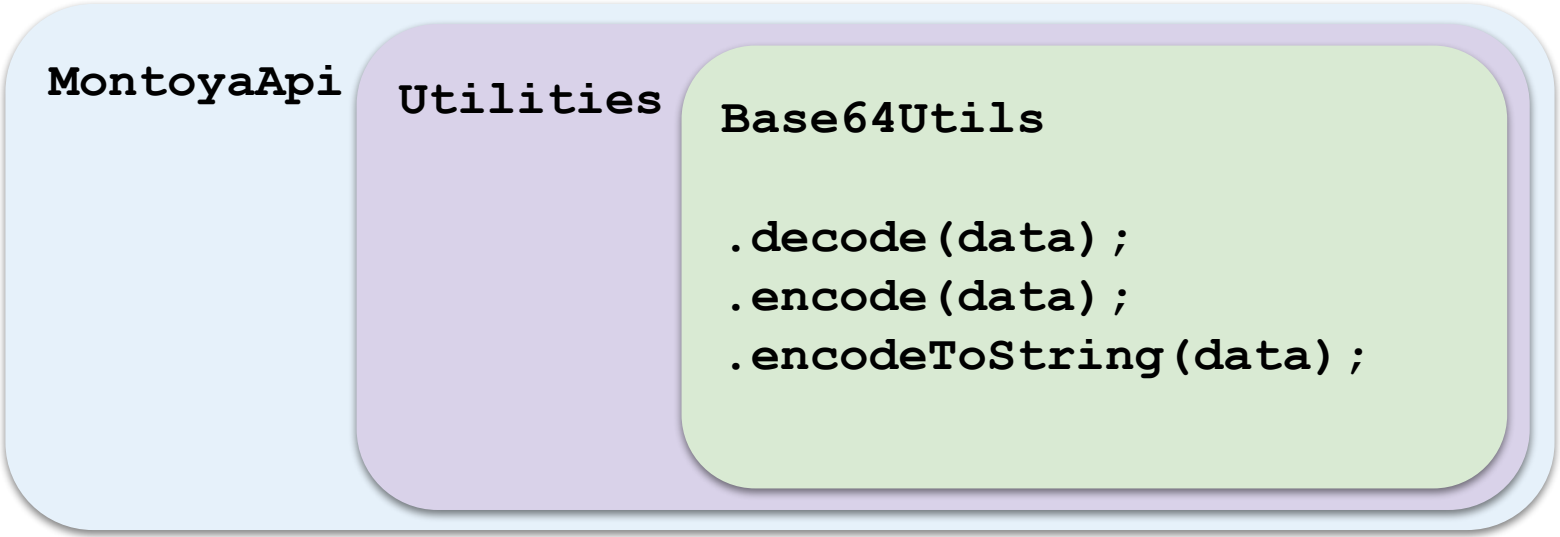
Logging capabilities




```
montoyaApi.logging().logToOutput(“Hello world!”);
```

```
logging.logToOutput(“Hello world!”);
```

Utilities




```
montoyaApi.utilities().base64utils().encode("Hello world!");
```

```
montoyaApi.utilities().base64utils().decode("SGVsbG8gd29ybGQh");
```

How to write a Bambda

When might you need a Bambda



Cut through the noise



Speak its language



Answers in seconds

How Burp runs your Bambda

Your Bambda is the content of a method that will be called by Burp when it is run - a fragment of Java code

```
return_type function_name(Parameter parameter1) {
    <YOUR_CODE_FRAGMENT>
}
```

Methods have a return type - the object you need to give back at the end of the method

You'll be given parameters & references to Montoya objects to use

return object type

expand for more info

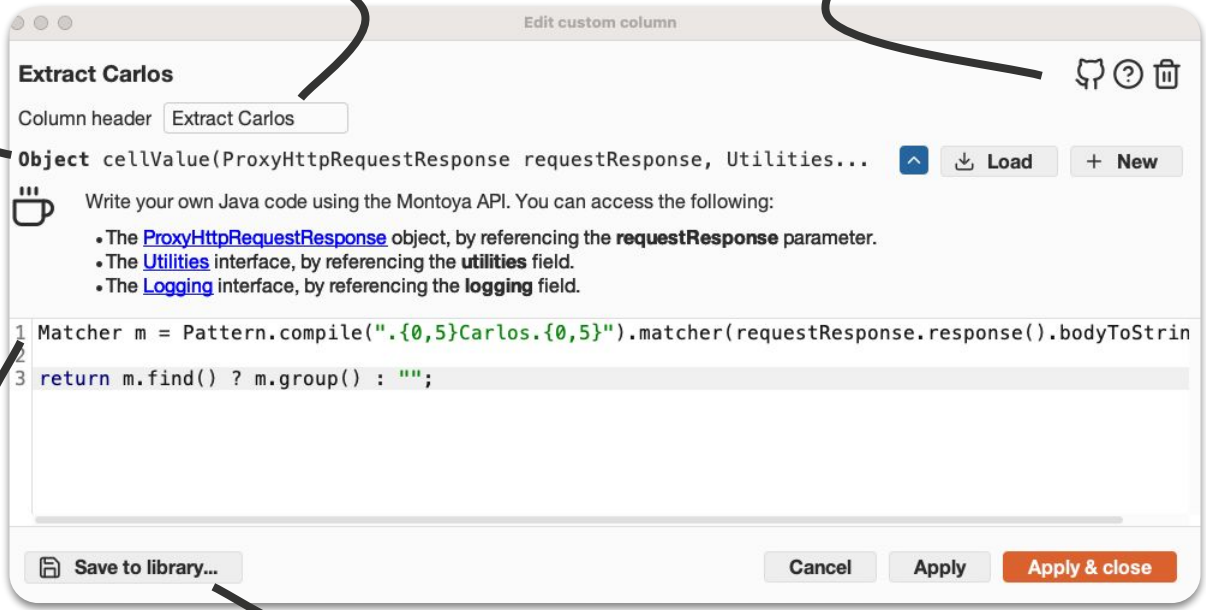


objects and references you can use

code editor with autocomplete

readable column name

direct links to github and documentation



return object type - could be string, icon, boolean

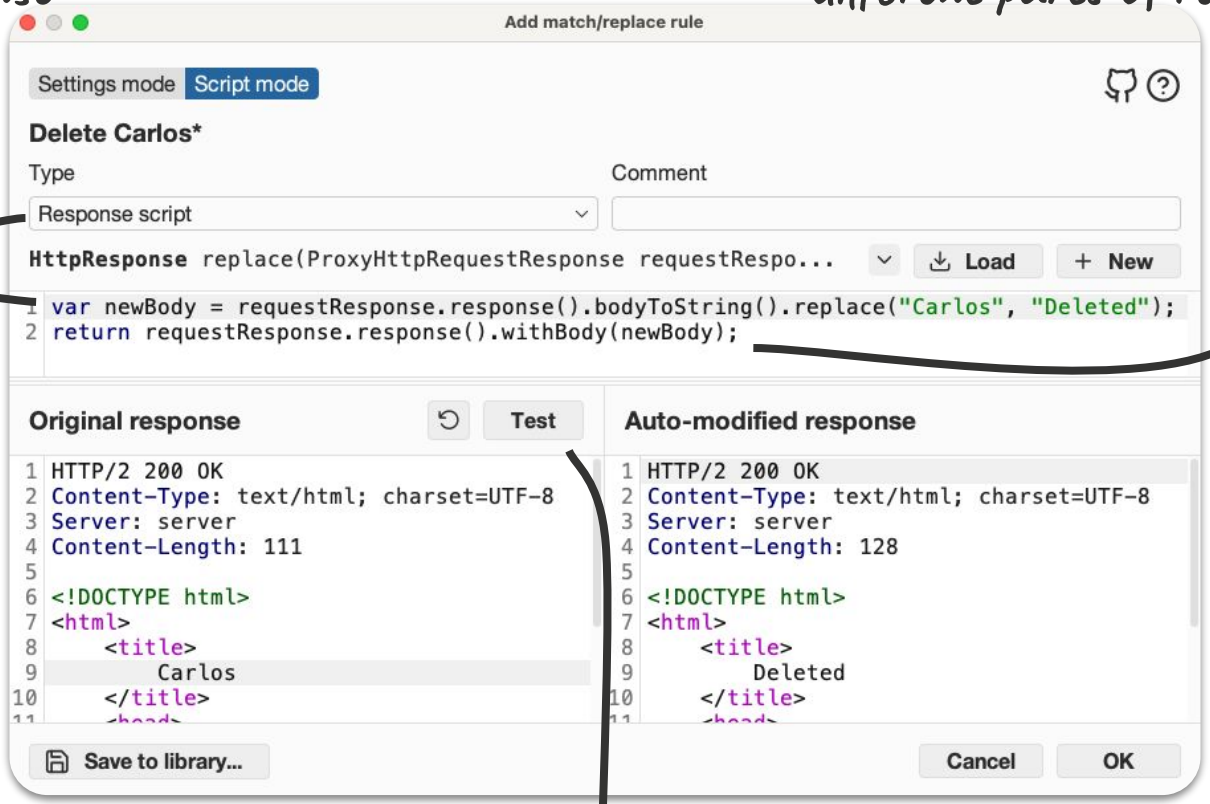
built-in java utilities

save your bambda to your bambda library for reuse

select whether to act on request or response

montoya helper methods let you quickly update different parts of requests/responses

don't worry too much about types, just use var (variable)



built-in testing tool

worksheet

Solve with a Bambda

Home
Downloads & Resources
Programming Concepts
Bambda Workshop
Exercise A: Proxy History Intelligence
Exercise B: JWT Triage
Exercise C: Custom Scan Checks
Extension Workshop
Demo targets
Free Build Time

Bambda Workshop

Hands-on exercises for the *Burp, But Yours* workshop.

Write Bambdas that solve real proxy-history problems, then test them against Web Security Academy labs.

TABLE OF CONTENTS

1 What is a Bambda?
2 Loading a Bambda
3 Select an exercise
   a Exercise A: Proxy History Intelligence BEGINNER
   b Exercise B: JWT Triage INTERMEDIATE
   c Exercise C: Custom Scan Checks ADVANCED

What is a Bambda?

Bambdas are Java lambda expressions that run inline in Burp Suite. The types are:

- Table filters: control which rows appear in proxy history, logger, or site

This site uses Just the Docs, a documentation theme for Jekyll.

Bambda workshop

send us feedback

EXERCISE A

Proxy history intelligence

Filters, custom columns

EXERCISE B

JWT triage

Filters, custom columns, string manipulation

EXERCISE C

Custom scan checks

Passive checks, active checks, insertion-point request building

- portswigger.github.io/burp-but-yours

hands up...

Which exercise did you pick?

Did you complete it?

Did anyone manage to do the “Taking it further” exercises?

How to write an extension

When might you need an extension



Eliminate repetitive manual steps



Custom protocols, auth, and encoding



Something the community reaches for

Extension starter project

You could start a project from scratch, but a starter project is available to reduce setup

- **Extensions > APIs > Download starter project**

- https://github.com/PortSwigger/extension-template-project
  Clicking “Use this template” will provide a copy of this project in your own GitHub account with the name that you want

Extension starter project

File structure overview

```
extension-template-project/
  ├── CLAUDE.md
  ├── README.md
  ├── build.gradle.kts
  ├── docs/
  ├── gradle/
  └── src/
      └── main/
          └── java/
              └── Extension.java
```

What's what?

`CLAUDE.md` and `docs/` - agentic coding development

`gradle` files - building your extension `JAR` file

`build.gradle.kts` - libraries and dependencies

`src/main/java/Extension.java` - code entry point

Loading an extension into Burp

1. Build your extension into a JAR.

2. Go to **Extensions > Installed > Add** and select your JAR file.

3. Select the Auto-reload checkbox.

Burp now reloads your extension every time you rebuild the JAR, so you don't need to reload it by hand.

Avoid starting from scratch

Build from existing work where possible

Example extensions act as a good base to extend and adapt

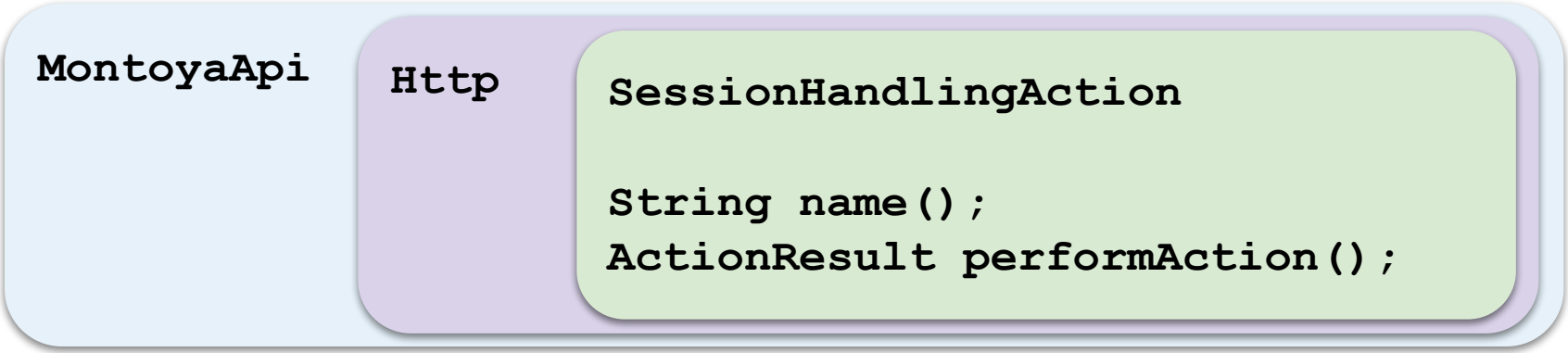
- PortSwigger/burp-extensions-montoya-api-examples

BApp Store extension code is available on the PortSwigger GitHub

Registrations

Registrations are your integration point into Burp

Use these to hook into different parts of Burp




```
montoyaApi.http().registerSessionHandlingAction(...);
```

Implementing interfaces

An interface describes the methods a class must implement

You provide the implementation for each required method

Implemented methods are marked with `@Override`, showing that you're providing the interface implementation

Burp will call the method when triggered, and your code will run

First steps

1. Plan: What is the problem you are trying to solve?
2. Identify: Which parts of your code “hook” into Burp?
3. Code: Discuss with coding agent, or implement registrations

worksheet

Build / adapt an extension

Home
Downloads & Resources
Programming Concepts
Bambda Workshop
Extension Workshop
Exercise A: Custom Editor Tab
Exercise B: Custom Scan Check
Exercise C: Session Handling
Demo targets
Free Build Time

Extension Workshop

Hands-on exercises for the *Burp, But Yours* workshop, where you build a working extension using the Montoya API.

TABLE OF CONTENTS

1 Prerequisites
2 The Montoya API
3 Set up your development environment
   a 1. Get the starter project
   b 2. Open it in IntelliJ
   c 3. Build the JAR
   d 4. Load it into Burp
4 Working with classes
   a One public class, one file
   b Implementing an interface
   c Why split the work across classes
   d IntelliJ shortcuts
5 The development loop

This site uses Just the Docs, a documentation theme for Jekyll.

Extension workshop



EXERCISE A

Additional editor tabs

Conditional logic, string manipulation

EXERCISE B

Custom scan checks

Passive checks, active checks, insertion-point request building

EXERCISE C

Handling custom authentication with sessions

Request mutation, third-party libraries

- portswigger.github.io/burp-but-yours

hands up...

Which exercise did you pick?

Did you complete it?

Did anyone manage to do the “Taking it further” exercises?

free dev

Build your own prototype

send us feedback

YOU SAID EARLIER

“What's one repetitive manual testing task you'd automate?”



Go!

No ideas? See “Free Build Time” on the workshop site

https://portswigger.github.io/burp-but-yours/

call out...

What did you build?

What problem were you solving?

How far did you get?

Sharing your work

Share your work with the community

BCheck Scan Checks

PortSwigger/BChecks

Open a pull request

Bambda Scripts

PortSwigger/bambdas

Open a pull request

Sharing extensions

Via GitHub

Share your work and get feedback directly from users of your extension

BApp Store

PortSwigger/extension-portal

Raise an issue

BApp Store criteria

1. Submissions go through a review process
2. Must provide unique capabilities
3. Code must meet our distribution standards

Full criteria available on the PortSwigger website



Join the conversation

Share what you built, ask questions, and see what the rest of the community is making

discord.gg/portswigger

Wrap-up

TODAY YOU

Explored Bambdas and extensions

Built and tested your own working examples

Learned where to share what you make

KEEP GOING

Finish your prototype

Chat about it on Discord

Open a pull request or submit to the BApp Store

https://forms.gle/qHvG7JMqnL3EXtf96



Any questions?

https://discord.gg/portswigger

PortSwigger

